KIP-15 - Add a close method with a timeout in the producer

Status

Current state: *Accepted*

Discussion thread: here

JIRA: *KAFKA-1660*, KAFKA-1659

Released: 0.8.3

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Current KafkaProducer.close() method will try to finish sending all pending messages before it returns. There are several motivations to add a close method with timeout in the producer.

1. Sometimes, user will want to close a producer within a bounded time to avoid blocking on producer.close() for too long.
2. One specific use case of 1) is that in some scenarios, user will want to close the produce immediately and fail all the unsent messages in RecordAccumulator. Some examples are:
    a. In mirror maker, if a send failed, we don't want to continue sending messages in RecordAccumulator to avoid reordering.
    b. For people who are using deployment tools, a service is expected to stop in given time. In that case people might want to have a bounded time to shutdown producer.

So we need to provide an interface that allow user to choose to close producer in which way.

Public Interfaces

Add another interface:

public void close(long timeout, TimeUnit timeUnit)

This call waits some specified time for the producer to close. If the producer did not finish closing before timeout, it close the producer forcefully by failing all the unsent messages.

- close(0, TimeUnit.MILLISECONDS) - do not wait, fail all the incomplete requests and close the producer.
- close(10000, TimeUnit.MILLISECONDS) - wait at most 10 seconds for the producer to send pending messages, if messages cannot be sent in 10 seconds, let the sender thread fail the rest of messages, wait for sender thread to complete and close producer.
- close(-1, TimeUnit.MILLISECONDS) - IllegalArgumentException will be thrown.

Several addition considerations:

- The close(timeout) should work when called from a user caller thread or the internal sender thread.
- In sender thread only close(0, TimeUnit.MILLISECONDS) should be called because:
    - close() should not be called because that will make producer block forever.
    - close(500, TimeUnit.MILLISECONDS) has the same effect as close(0, TimeUnit.MILLISECONDS) only except it will block for 500 milliseconds.
- **IMPORTANT, for the reason above, if a close() or close(timeout, TimeUnit) is called from sender thread (i.e. callback). An error message will be logged and close(0, TimeUnit.MILLISECONDS) will be called instead.**
- Because it is possible the close(timeout) is called for multiple times, it has to be idempotent.

Proposed Changes

To close a producer forcefully, the following changes are going to be made:

1. Add a forceClose flag to sender.
2. When forceClose flag is set, sender will not send any more messages but fail all the messages in RecordAccumulator and wake up the threads waiting on a callback. These threads may be doing:
    a. synchronized send
    b. flush()

3. Cleanup metrics and release other resources.

When close(positive, TimeUnit.MILLISECONDS) is called, it will try to do a normal close first. If the normal close did not finish before timeout, it then close the producer forcefully. It will also wait the sender thread to finish.

If user calls close() in callback, it will block forever because the sender thread will try to join itself. So if a close() call is called from sender thread, **an error message will be put in the log and close(0) will be called instead**.

Compatibility, Deprecation, and Migration Plan

This is a backward compatible change.

Rejected Alternatives

Add an abort() call which close producer forcefully. Leave the close() as is (KAFKA-1659)This is a viable solution but does not provide easy solution to a time bounded shutdown. As quoting from KAFKA-1660 commented by Andrew Stein.

> " I don't think that this (the request for producer.close(timeout)) and KAFKA-1659 (the request for producer.abort()) are the same.
>
> If producer.close(timeout) were to return without actually closing the producer in time, the user could then continue with producer.abort(). However, this would not be the only case where producer.abort() would be useful. It could be used if a producer.send() were to fail or if a future.get() were to fail as well.
>
> In addition if producer.close() were to return without actually closing the producer in time, the user could continue with other logic besides producer.abort(). "

In KAFKA-1659, we decided to merge the abort() into producer.close(timeout) to avoid adding another interface. It is also agreed in KAFKA-1934.